SENG 466 PROJECT #2 DESIGN REPORT

SOFTWARE AND ELECTRICAL DESIGNS

Authors: Radost Rychtera, V006885306
Bruno Sampaio, V00792369
Mike Sykes, V00639903

Date: 19 April 2013
Total Pages: 15

Table of Contents

Table of Figures

1 INTRODUCTION

Project 2 of SENG 466: Software for Embedded and Mechatronic Systems course was to design a piano playing robot in conjunction with the Music and ECE Masters Departments at the University of Victoria. The task for this project is to design and build a robotic instrument which can play piano keys at various speeds and for varying periods of time. The idea of this project is to be a proof of concept for a robot that can play one key and an array of the device could potentially play a full keyboard. The instrument can be preprogrammed to play the note in a desired way. To accomplish this task a support structure was designed and built to support the solenoid mechanism which would play the note, a solenoid was designed to effectively play the note, and an embedded system involving microcontroller was designed to drive the solenoid.

This report will outline the electrical design behind the solenoid construction and the computing developments to drive it.

2 SOLENOID DESIGN

An electrically activated, linear actuator used to create linear motion in a system is called a solenoid. The basic concept behind the solenoid is a long thin wire wrapped around a magnetic material to enhance a magnetic field.

The solenoid is comprised of four main components: a tightly wound coil of wire wrapped around a cylindrical structure, a plunger called an armature made of a magnetic material, and a spring to return the armature.

Solenoids can be broken down into two categories for either pushing or pulling a load. This report focuses on a pushing type of solenoid used to apply a downward force to a piano key.

2.1 CHARACTERISTICS OF A SOLENOID

A solenoid works by creating a magnetic field (**B**) within a current carrying coil of wire as seen in **Figure 1** below. Inserting a ferromagnetic object, such as iron or steel,

causes the object to get move towards the center of the coil. Increasing the number of turns of wire, and/or increasing the current through the coil, increases the magnetic field strength and also the total mass the solenoid can actuate.

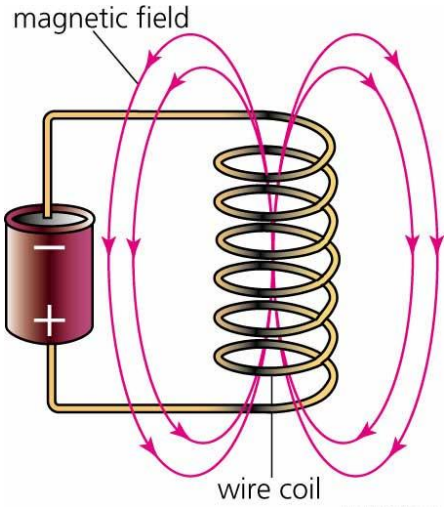

Figure 1 Simple Magnetic Field

The **B** field in a solenoid is calculated from the equation below and shows the relationship between current and number of turns in the coil.

$$B = \frac{\mu_0 NI}{2R}$$

Where R = Radius of coil [units^2*π]

I = Current through coil [Amps]

N = Number of turns of wire [Integer]

To determine the force applied from a solenoid when a coil is filled and partially filled with a ferromagnetic material, the energy equations can be employed as:

$$F = \frac{\Delta U}{L}$$

$$\Delta U = U_{filled} - \left(U_{partfill} + U_{partnotfill}\right)$$

Where L = length of coil [units]

U = energy of coil

F = Force [N]

To determine the energy created by a coil partly filled with a ferromagnetic material, two equations must be employed. One equation finds the energy of the empty side of the coil, while another finds the energy of the filled part of the coil as displayed below.

$$U_{filled} = \frac{VB^2\mu_m}{2\mu_o^2}$$

$$U_{partfill} = \frac{B^2(L_1\pi R^2)\mu_m}{2\mu_0^2}$$

$$U_{partnotfill} = \frac{L_2\pi R^2 B^2}{2\mu_0}$$

Where B = (NI)

μ_m=magnetic permeability of material

μ_0 = permeability of free space

L = length of coil

L_1 = length of material inserted in coil

L_2 = air gap length

V = Volume = $L\pi R^2$

Since the relation of **B** field is dependant to the number of turns and the current through the turns, each gauge of wire will have a maximum limit of power due to the intrinsic resistance of copper wire. Having more turns results in a higher resistance and

a lower current. Increasing the voltage across the coil increases the current through the turns and thus increases the power dissipated from the coil as heat.

A higher resistance is the product of more turns and results in higher heat dissipation in the coil. Because the windings are created with an enamel coated copper wire that is easily melted with heat, a solenoid must be designed with wire gauge and power requirements in mind.

2.2 Initial Design Attempts

The first version of the solenoid design attempt involved a 3-D printed plastic solenoid base wound with N=1614 turns of 28 AWG copper enamel wire. The solenoid was tested to a current maximum of I=7.88A at 20 volts. The solenoid got very hot and dissipated too much power from high resistance. Since the solenoid could not produce enough force to actuate the piano key, a larger gauge of wire was required to limit the resistance to increase the magnetic field.

Figure 2 Initial solenoid attempt

2.3 FINAL DESIGN

The last design of the solenoid was created with a square, plastic stock base wound with 20 AWG enamel coated copper wire as seen in Figure 3. The coil has N=512 turns and draws a current of I=8A at 7.2V to actuate a steel armature attached to a wooden dowel and drive the piano key.

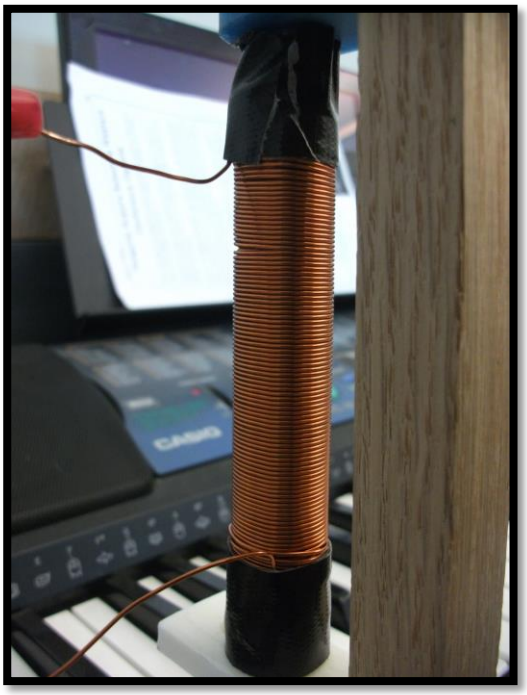

Figure 3 Final Solenoid Design

3 SOFTWARE AND HARDWARE DESIGN

The system of a robot playing piano was built using the Seeeduino Mega, a version of Arduino Mega which contains the ATmega 1280 chip. Some features of this board are selectable 5V/3.3V operation, 70 digital IO pins, 16 analog inputs, 14 PWM outputs, 4 hardware serial ports (UART), compatible with most Duemilanove and Diecimila shields, smaller size, can be powered using a battery or through a AC to DC adaptor.

Table 1 - Specifications

Operating Voltage	5V/3.3V
Operating Temperature	-20~70℃

Input Voltage	6-20V
DC Current per IO pin	40mA
Flash Memory	128k
SRAM	8k
EEPROM	4k

The next figure shows the board used for this purporse.

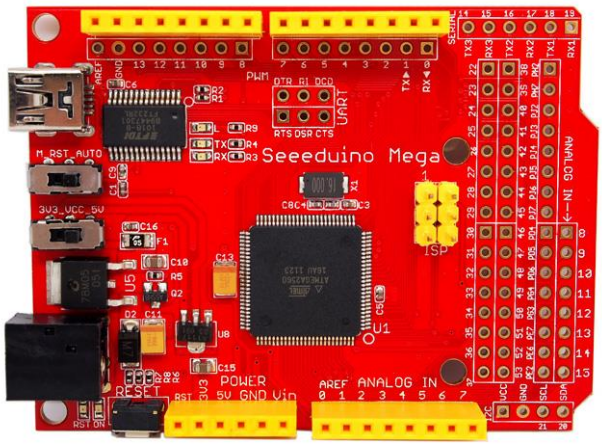

Figure 4 - Seeeduino Board

The system drives a relay which is responsible for providing the current and voltage for solenoid which will be described in the next section.

3.1 RELAYS

In order to drive the solenoid, a relay was chosen to provide an ideal separation between our load to be controlled and our control system. They are used where it is necessary to control another circuit from a low-power signal (with electrical isolation between control and controlled circuits). This is considered a safe way to drive this kind of loads that need high current or voltage and avoid damage on the microcontroller part of the system.

A relay is an electrical device operated by switch. They are composed by electromagnet, armature, spring, and contacts. They have two circuit compositions, the

first involves the electromagnet and the second is related to armatures that will be connected to some output load.

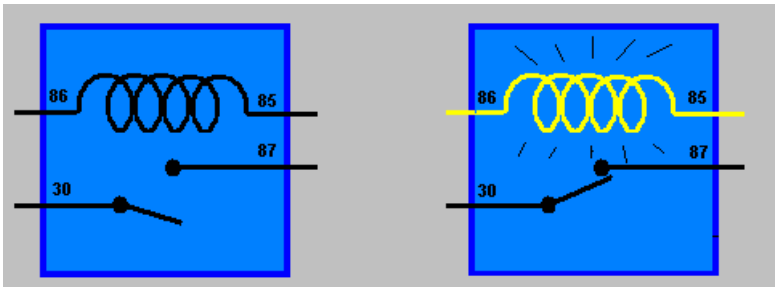

Figure 5 - Relay

In this picture, there are contacts to control power to the electromagnet. When the electromagnet is energized, it attracts the armature and the signal is conducted through contacts 30 and 87 on the picture (characterizing the “on” state). Basically, this armature is used like a switch in the second circuit. On the other hand, when the electromagnet is not energized, the spring pulls the armature back and the circuit is not close (it characterizes the “off” state). Usually the contacts have special names because of their initial states. The contact number 87 is initially disconnected; its name is normally open (NO). There is another contact right below the contact number 87 on the figure (it is not drawn) that is initially connected to contact 30, its name is normally closed (NC).

Some important features must be analyzed when looking for relays. Each one has the minimum and maximum voltage and current that is needed to activate the armature and to support loads. These devices commonly need more than 50mA to be activated, that means, most kinds of microcontrollers do not provide this amount of current. In the case of ATmega 1280 used in this project, the maximum DC current provided is 40mA, thereby, the designer cannot plug an I/O pin directly to energize the relay because it will require much more current than microcontroller can do and, therefore, it will burn the pin. To overcome this problem, a circuit using bipolar junction transistor was applied and will be discussed in the next sections.

Relays have many applications; they are used in home automation to turning on motors, lights, etc. They are used to start a motor vehicle as well.

3.2 USING RELAY WITH MICROCONTROLLERS

To effectively drive a relay from a microcontroller, it is necessary to connect a bipolar junction transistor to provide a gain of current necessary to energize the electromagnet inside the relay. Connecting the contacts of it through a collector section of the BJT will support the necessary current. It is common to use a diode in parallel to the load just in case to absorb sparks from the output circuit. The "pin" is a usual connection from any microcontroller used to drive. On the next picture it is possible to see the schematic with a simple application to turn the led on using external power source. As most applications, high level power, it is necessary to couple the load with its enough power supply and use relay to actuate it.

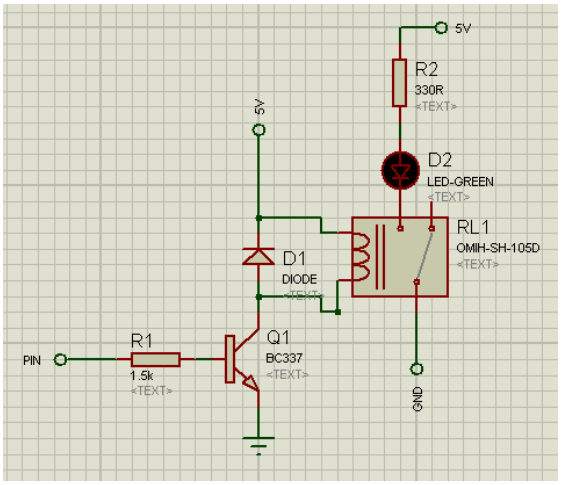

Figure 6 - Circuit to drive relay

For this project, a relay shield was used. It provides 4 relay connections. Each one has the circuit above behind each relay device.

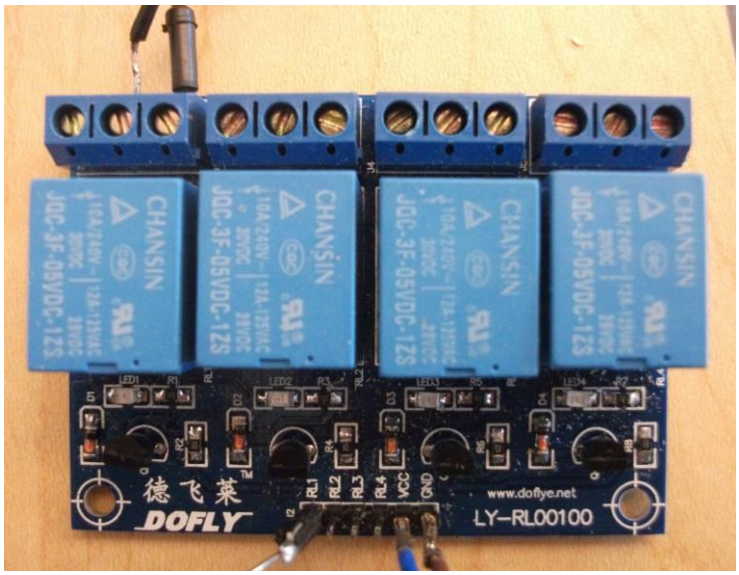

Figure 7 - Relay pack

3.3 DRIVING THE SOLENOID

To drive the solenoid, the function of PWM (Pulse Width Modulation) provided by microcontroller was used. Because of heating issues when energized, the solenoid active time must be minimized and this is achieved by sending a PWM signal and setting the percentage of active signal the designer want to turn on your device.

The function analogWrite() included in standard Arduino library can be used to generate a steady square wave of specified duty cycle and it remains the signal until when next call to analogwrite() occur. The default frequency of this signal is 490Hz. The parameters of this function are the pin in the board and the duty cycle value between 0 (completely off) and 255 (100% on). In our case, the 4-relays pack is activated on low level, it is inverted probably for safety purposes. So, our firmware was developed using the minimal "off" period of duty cycle.

4 CONCLUSION AND RECOMMENDATIONS

The aim of this project was to design and build a self-actuated piano playing robot. Due to time constraints the robot designed was only proof of concept and can be applied on a larger stage with an array of solenoids. This report discussed the issues which arose in the design and building of this robot. Due to changes in the solenoid deign, two

stands were designed both with unique designs. The solenoid bobbin design was iterative as the first printed design was found to be too weak to support the coil. Overall the mechanical design was pretty straight forward and only required changes due to restrictions in the manufacturing process.

A sleeker and more accurate design could have been obtained if the stand was designed with aluminum parts however due to time constraints it was found to be better to used materials available in the lab and the 3D printer.

APPENDIX A

```
/*
  Group: Bruno Sampaio, Mike Sykes, Radost Rychtera
  Projet 2: Robot piano

Pins 9 and 10: controlled by timer 1

Setting       Divisor       Frequency
0x01          1             31250
0x02          8             3906.25
0x03          64            488.28125
0x04          256           122.0703125
0x05          1024          30.517578125

TCCR1B = TCCR1B & 0b11111000 | <setting>;

25% -> 64
50% -> 127 OK!
55% -> 140 NO
60% -> 153 NO
75% -> 191 NO
100 -> 255

*/

#define PIN_11_MEGA 13

void turnOn () {
  analogWrite(PIN_11_MEGA, 127);
}

void turnOff () {
  analogWrite(PIN_11_MEGA, 255);
}

// hold the key for 'ms' milliseconds
void play(unsigned int ms) {
  turnOn();
  delay(ms);
  turnOff();
}

void setup ()
{
  // freq ~= 30Hz
  TCCR1B = TCCR1B & 0b11111000 | 0x05;
  pinMode(PIN_11_MEGA, OUTPUT);
}

// example code to play a simple melody using the same note
void loop ()
{
  while(true) {
    play(50);
    delay(250);
    play(1000);
    delay(700);
    play(50);
    delay(700);
    play(50);
```

```
    delay(700);
    play(50);
    delay(700);
    play(50);
    delay(700);
    delay(8000);
  }
}
```